**上机实验2 常用GIS功能的开发**

**【1.上机目的】**

掌握基于ArcEngine的GIS基本功能的开发方法。

**【2.上机内容】**

选用一种高级程序设计语言（如，C#、VB等）结合ArcGIS Engine开发一个包含GIS常用功能的应用程序，实现GIS基本操作：包括放大、缩小、漫游、全图显示、打开/保存Mxd地图文档功能、距离量算/面积量算；实现空间查询功能：包括点选查询、图形和属性数据互查[可选]、缓冲区查询/分析[可选]。（说明：可选内容二选一。）

**【3.上机要求】**

1．完成代码的编写工作，基于ArcGIS Engine实现GIS常用功能开发。

2．掌握利用组件式GIS技术开发应用型地理信息系统的基本思路。

3．独立实验，认真、按时完成上机实验报告。

**【4.上机步骤】**

实验完整代码连接：<https://github.com/mxxxOrz12/PyQGIS>

# 1 技术选型

由于ESRI公司在2014年左右就已经停止对ArcEngine的更新，推出了新的桌面开发包ArcGIS Runtime SDK，但由于其为商业软件，且国内相关应用于教程较少，因此本实验使用PyQGIS（QGIS+Qt+Python）进行桌面GIS开发。



图1 项目结构

# 2 窗体设计



图2 窗体设计

通过在Qt中设计界面，并将.ui文件转换为python文件运行，界面共有四个部分：功能区、图层树、图层显示、底部状态栏。

# 3 功能实现

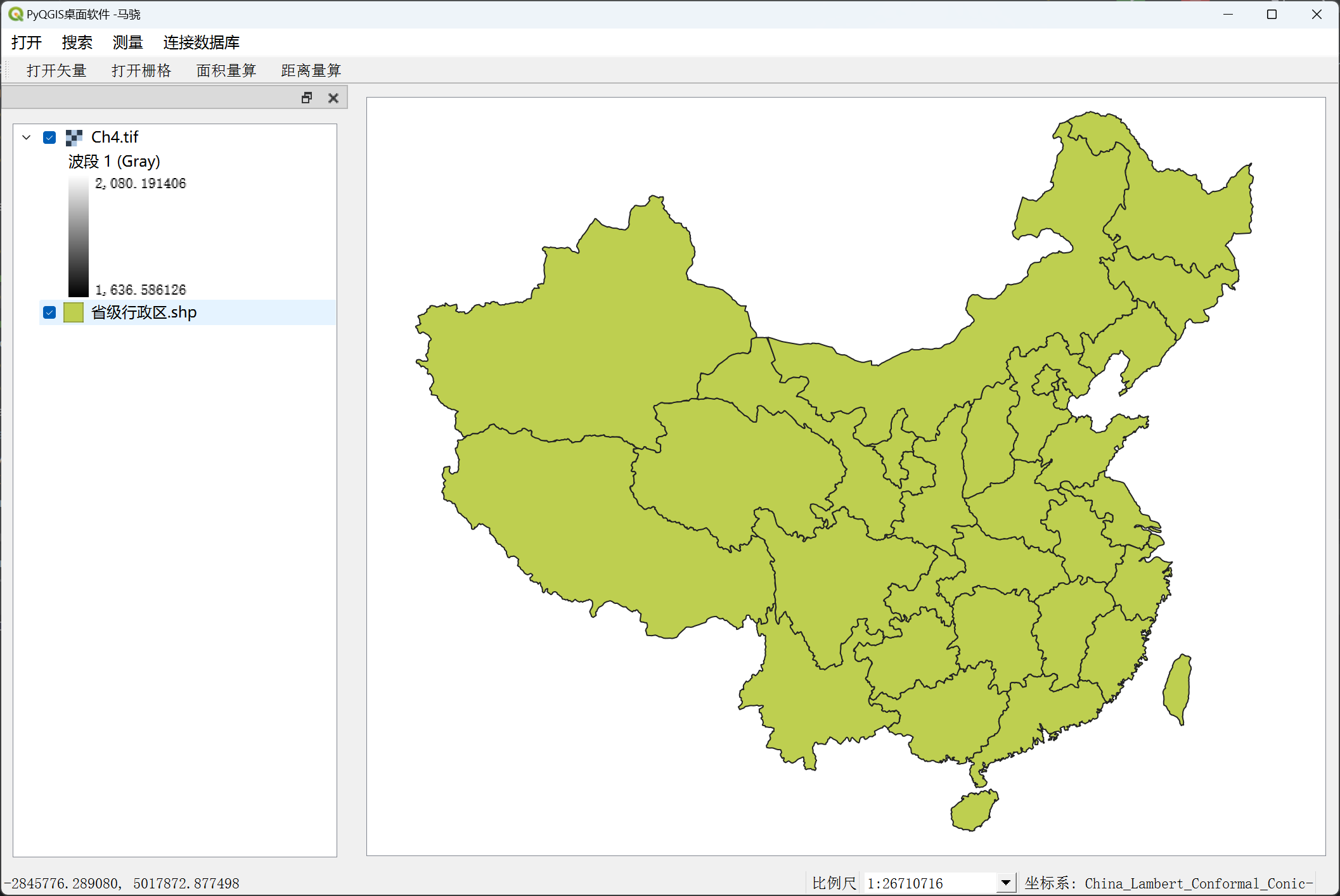


图3 加载矢量数据

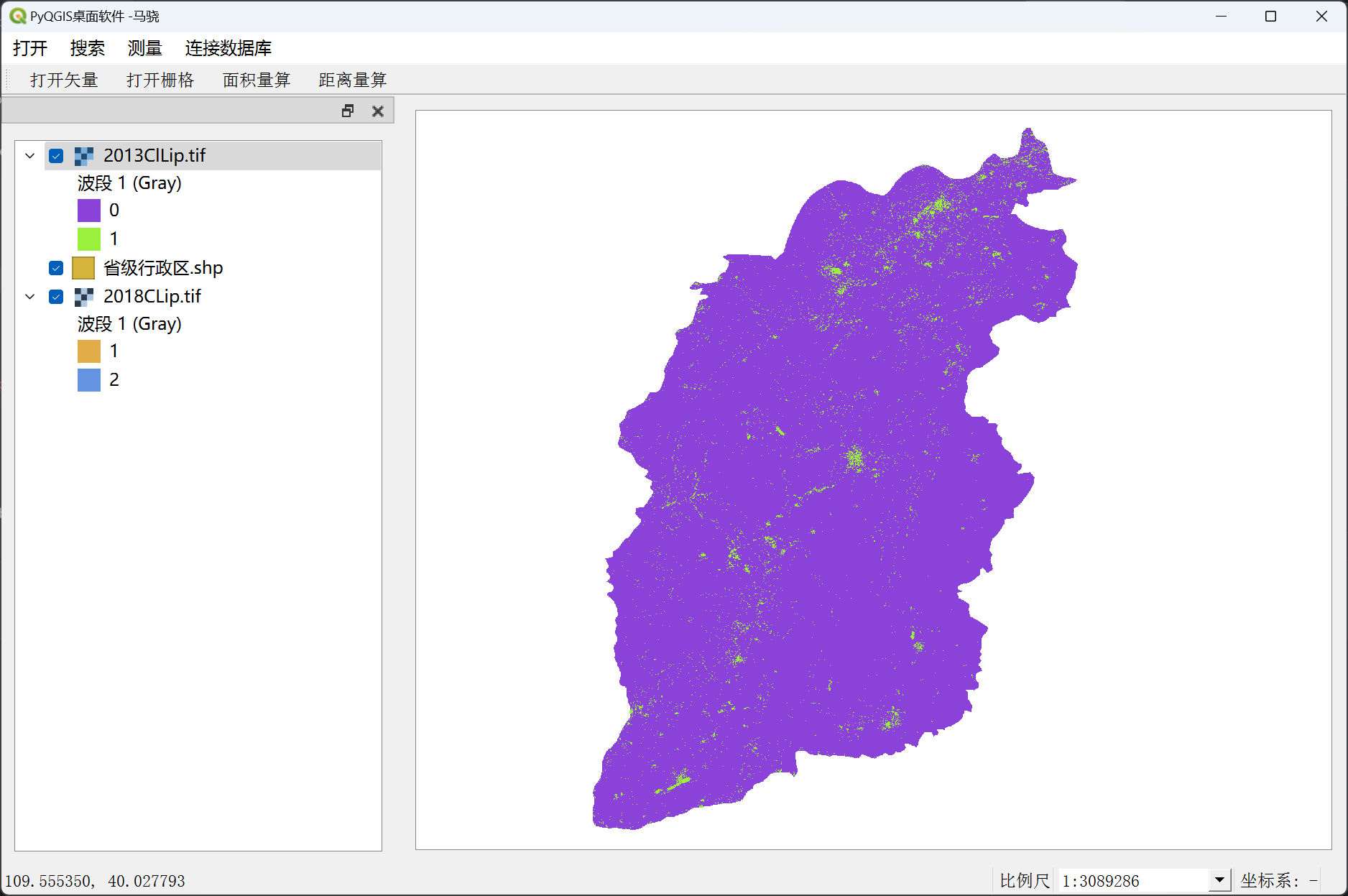


图4 加载栅格数据

代码如下，QGIS通过底层的GDAL来加载栅格和矢量数据，在main定义两个函数，addRasterLayer和addVectorLayer，并判断是否是第一次添加，如果是需设置图层显示框的坐标系与该数据的坐标系相同。

1. **def** addRasterLayer(self, rasterFilePath):
2. rasterLayer = readRasterFile(rasterFilePath)
3. **if** self.firstAdd:
4. addMapLayer(rasterLayer, self.mapCanvas, True)
5. self.firstAdd = False
6. **else**:
7. addMapLayer(rasterLayer, self.mapCanvas)
8. # 添加矢量图层
9. **def** addVectorLayer(self, vectorFilePath):
10. vectorLayer = readVectorFile(vectorFilePath)
11. **if** self.firstAdd:
12. addMapLayer(vectorLayer, self.mapCanvas, True)
13. self.firstAdd = False
14. **else**:
15. addMapLayer(vectorLayer, self.mapCanvas)

上面的两个函数中调用了qgisUtils文件夹下的addMapLayer、readRasterFile、readVectorFile函数，其中addMaplayer中我判断了是否为第一个添加的图层，并进行坐标系、显示范围的设置，并更新layers。

1. **def** addMapLayer(layer:QgsMapLayer,mapCanvas:QgsMapCanvas,firstAddLayer=False):
2. **if** layer.isValid():
3. **if** firstAddLayer:
4. mapCanvas.setDestinationCrs(layer.crs())
5. mapCanvas.setExtent(layer.extent())
6. **while**(PROJECT.mapLayersByName(layer.name())):
7. layer.setName(layer.name()+"\_1")
8. PROJECT.addMapLayer(layer)
9. layers = [layer] + [PROJECT.mapLayer(i) **for** i **in** PROJECT.mapLayers()]
10. mapCanvas.setLayers(layers)
11. mapCanvas.refresh()
12. # 读取栅格文件
13. **def** readRasterFile(rasterFilePath):
14. rasterLayer = QgsRasterLayer(rasterFilePath,osp.basename(rasterFilePath))
15. **return** rasterLayer
16. # 读取矢量文件
17. **def** readVectorFile(vectorFilePath):
18. vectorLayer = QgsVectorLayer(vectorFilePath,osp.basename(vectorFilePath),"ogr")
19. **return** vectorLayer

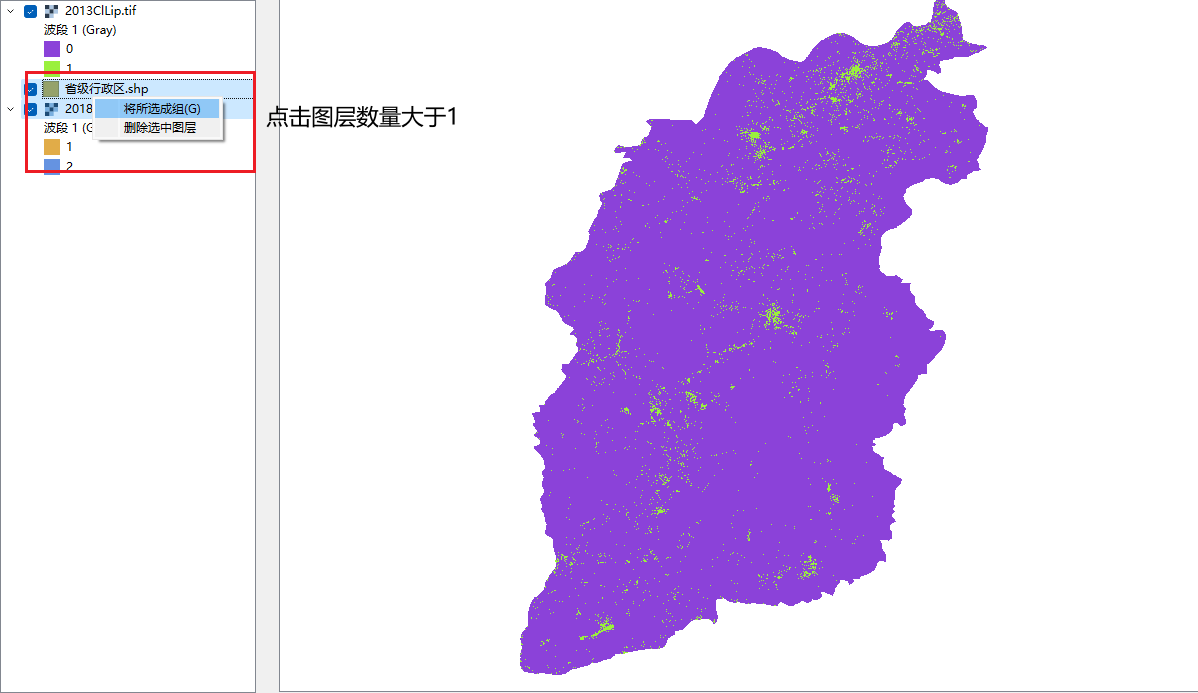


图5 右键图层事件绑定,点击图层大于1

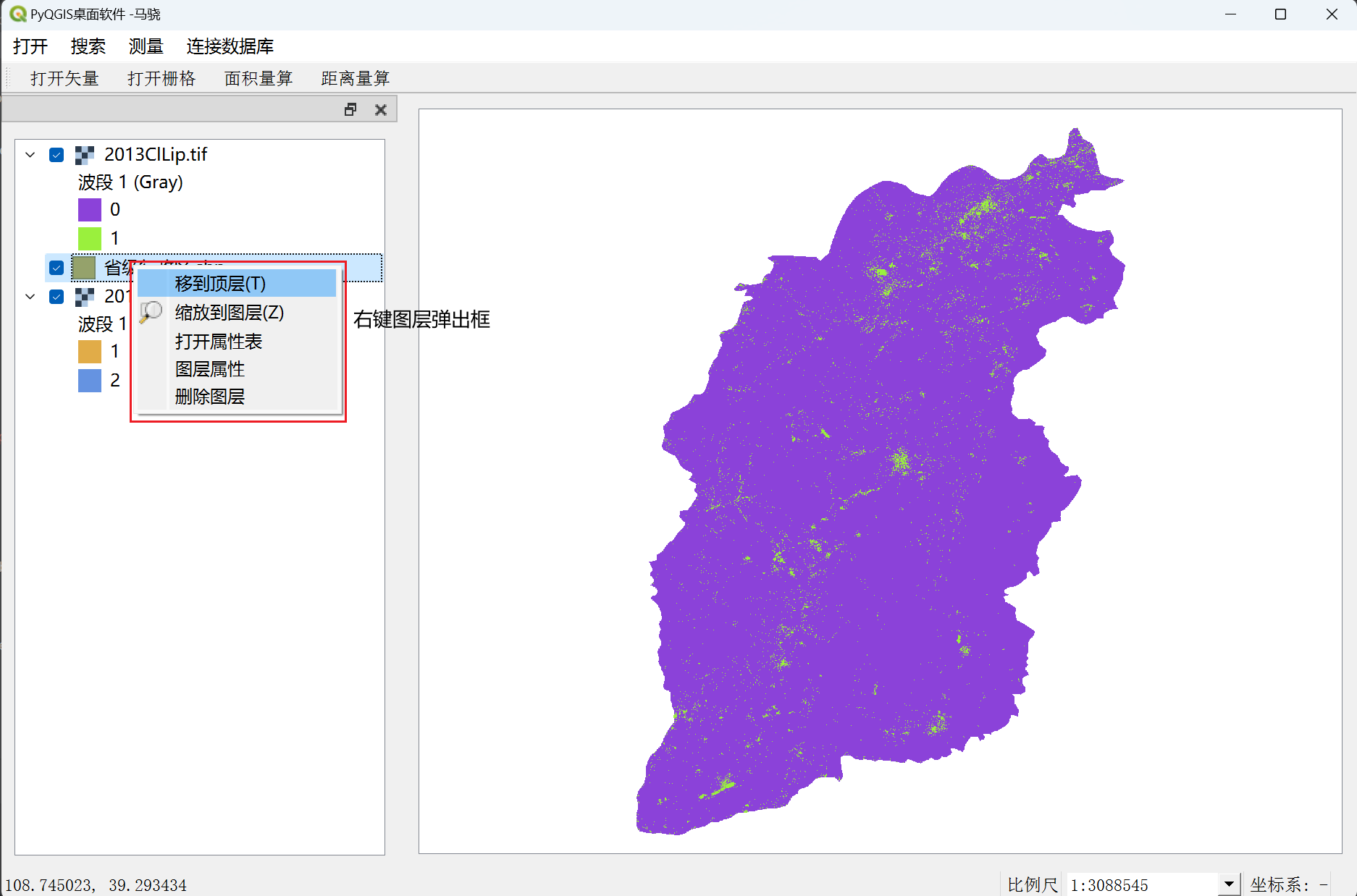


图6 右键图层事件绑定,点击1个图层

首先在qgisMenu文件中定义一个函数来创建QMenu对象，并且判断鼠标点击的图层数量，通过layerTreeView.selectedLayers()获取点击的图层数量，如果大于1，如图5所示添加合并为组和清除图层按钮。如果单击图层数量为1，则如图6，有移到顶层、缩放至图层、打开属性表、图层属性、删除图层等功能。

实现代码如下：

1. **def** createContextMenu(self) -> QtWidgets.QMenu:
2. **try**:
3. menu = QMenu()
4. self.actions : QgsLayerTreeViewDefaultActions = self.layerTreeView.defaultActions()
5. **if** **not** self.layerTreeView.currentIndex().isValid():
6. # 清除图层 deleteAllLayer
7. actionDeleteAllLayer = QAction('清除图层', menu)
8. actionDeleteAllLayer.triggered.connect(**lambda**: self.deleteAllLayer())
9. menu.addAction(actionDeleteAllLayer)
10. menu.addAction('展开所有图层',self.layerTreeView.expandAllNodes)
11. menu.addAction('折叠所有图层',self.layerTreeView.collapseAllNodes)
12. **return** menu
14. **if** len(self.layerTreeView.selectedLayers()) > 1:
15. # 添加组
16. self.actionGroupSelected = self.actions.actionGroupSelected()
17. menu.addAction(self.actionGroupSelected)
19. actionDeleteSelectedLayers = QAction('删除选中图层',menu)
20. actionDeleteSelectedLayers.triggered.connect(self.deleteSelectedLayer)
21. menu.addAction(actionDeleteSelectedLayers)
22. **return** menu
23. node: QgsLayerTreeNode = self.layerTreeView.currentNode()
24. **if** node:
25. **if** QgsLayerTree.isGroup(node):
26. group: QgsLayerTreeGroup = self.layerTreeView.currentGroupNode()
27. self.actionRenameGroup = self.actions.actionRenameGroupOrLayer(menu)
28. menu.addAction(self.actionRenameGroup)
29. actionDeleteGroup = QAction('删除组', menu)
30. actionDeleteGroup.triggered.connect(**lambda**: self.deleteGroup(group))
31. menu.addAction(actionDeleteGroup)
32. **elif** QgsLayerTree.isLayer(node):
33. self.actionMoveToTop = self.actions.actionMoveToTop(menu)
34. menu.addAction(self.actionMoveToTop)
35. self.actionZoomToLayer = self.actions.actionZoomToLayer(self.mapCanvas, menu)
36. menu.addAction(self.actionZoomToLayer)
38. layer: QgsMapLayer = self.layerTreeView.currentLayer()
40. **if** layer.type() == QgsMapLayerType.VectorLayer:
41. actionOpenAttributeDialog = QAction('打开属性表', menu)
42. actionOpenAttributeDialog.triggered.connect(**lambda**: self.openAttributeDialog(layer))
43. menu.addAction(actionOpenAttributeDialog)
45. actionOpenLayerProp = QAction('图层属性', menu)
46. actionOpenLayerProp.triggered.connect(**lambda** : self.openLayerPropTriggered(layer))
47. menu.addAction(actionOpenLayerProp)
49. actionDeleteLayer = QAction("删除图层",menu)
50. actionDeleteLayer.triggered.connect(**lambda** : self.deleteLayer(layer))
51. menu.addAction(actionDeleteLayer)
52. **return** menu
54. **except**:
55. **print**(traceback.format\_exc())

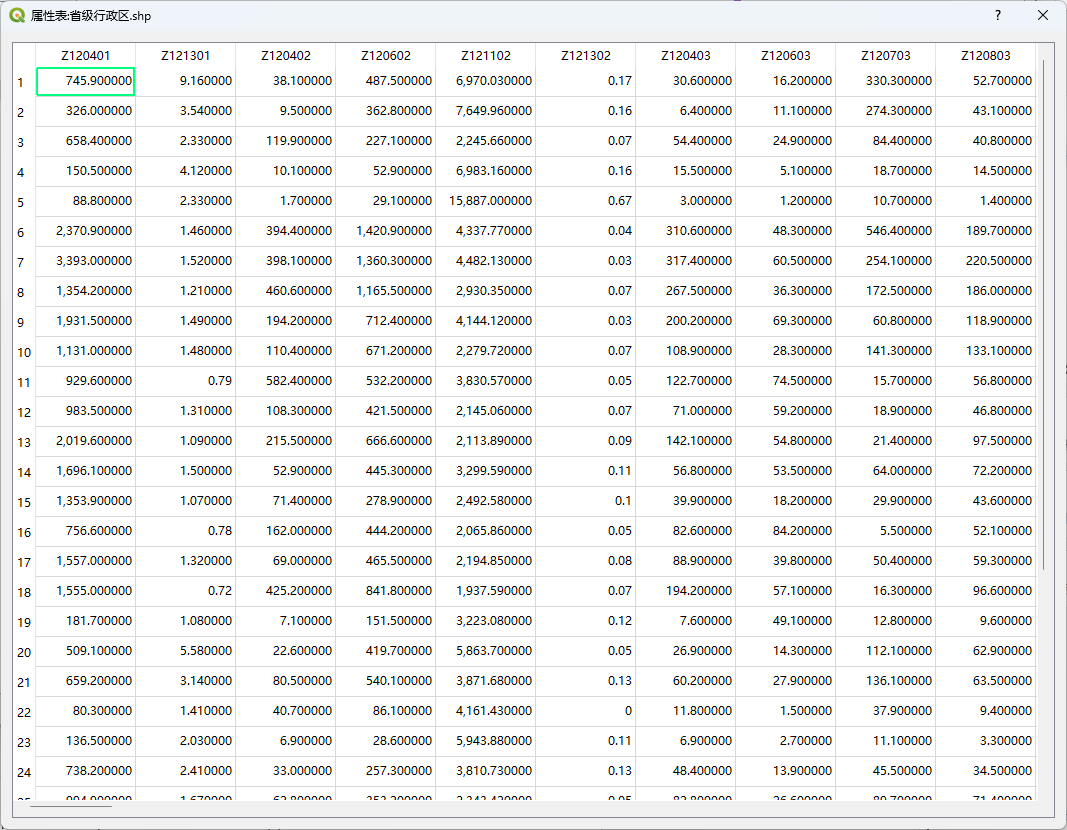


图7 图层属性表查看

属性表展示代码实现，首先创建一个AttributeDialog类继承QDialog，在openAttributeDialog函数中使用QgsVectorLayerCache缓存图层，使用QgsAttributeTableModel加载图层属性表，设置搭配tableView中进行加载。

1. **def** openAttributeDialog(self):
2. #iface
3. self.layerCache = QgsVectorLayerCache(self.layer, 10000)
4. self.tableModel = QgsAttributeTableModel(self.layerCache)
5. self.tableModel.loadLayer()
6. self.tableFilterModel = QgsAttributeTableFilterModel(self.mapCanvas, self.tableModel, parent=self.tableModel)
7. self.tableFilterModel.setFilterMode(QgsAttributeTableFilterModel.ShowAll)  #显示问题
8. self.tableView.setModel(self.tableFilterModel)

其它功能由于对QGIS文档不熟悉和Python能力不熟练开发耗时长，暂时没有实现。

**【5.上机体会】**

通过QGIS+Pyhton让我体会到了GIS开发真正的魅力，操作底层图层，点线面矢量编辑，图层属性等等，与WebGIS开发相比，开发时间较长，难度较大，且界面不美观。但现在桌面开发也有多种组件库，如PyQt-Fluent-Widgets，通过组件库能够快速开发美观的界面。

在实现GIS功能时，与ArcEngine集成开发相比QGIS更加底层，但更加麻烦，对开发者的编程能力和GIS能力需求较高，我只开发了较为简单的基础功能，代码量就达到了2000行左右，可见开发较为困难，但同时也十分灵活。希望以后能有机会通过QGIS开发提升自己GIS功底。